

The quadratic balanced optimization problem*

Abraham P. Punnen^{†1}, Sara Taghipour^{‡1}, Daniel Karapetyan^{§1}, and Bishnu Bhattacharyya^{¶2}

¹*Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia, V3T 0A3, Canada*

²*Google, Mountain View, CA, USA*

Abstract

We introduce the quadratic balanced optimization problem (QBOP) which can be used to model equitable distribution of resources with pairwise interaction. QBOP is strongly NP-hard even if the family of feasible solutions have a very simple structure. Several general purpose exact and heuristic algorithms are presented. Results of extensive computational experiments are reported using randomly generated quadratic knapsack problems as the test bed. These results illustrate the efficacy of our exact and heuristic algorithms. We also show that when the cost matrix is specially structured, QBOP can be solved as a sequence of linear balanced optimization problems. As a consequence, we have several polynomially solvable cases of QBOP.

Keywords: combinatorial optimization, balanced optimization, knapsack problem, bottleneck problems, heuristics.

1 Introduction

Let $E = \{1, 2, \dots, m\}$ be a finite set and \mathcal{F} be a family of non-empty subsets of E . It is assumed that \mathcal{F} is represented in a compact form of size polynomial in m without explicitly listing its elements. For each $(i, j) \in E \times E$, a cost c_{ij} is prescribed. The elements of \mathcal{F} are called feasible solutions and the $m \times m$ matrix $C = (c_{ij})$ is called the *cost matrix*. Then the *quadratic balanced optimization problem* (QBOP) is to find $S \in \mathcal{F}$ such that

$$\max\{c_{ij} : (i, j) \in S \times S\} - \min\{c_{ij} : (i, j) \in S \times S\}$$

is as small as possible.

*This research work was supported by an NSERC Discovery grant awarded to Abraham P Punnen.

[†]apunnen@sfu.ca

[‡]sara_taghipour@sfu.ca

[§]daniel.karapetyan@gmail.com

[¶]bishnu@gmail.com

QBOP is closely related to the *balanced optimization problem* introduced by Martello et al [21]. To emphasize the difference between the balanced optimization problem of [21] and QBOP, we call the former a *linear balanced optimization problem* (LBOP). Special cases of LBOP were studied by many authors [10, 13, 14, 17, 18, 21, 23]. Optimization problems with objective functions similar to that of LBOP have been studied by Zeitln [34] for resource allocation, by Gupta and Sen [16], Liao and Huang [20], and Tegez and Vlach [29, 30, 31] for machine scheduling, by Ahuja [2] for linear programming, and by Scutellà [28] for network flows. A generalization of LBOP where elements of E are categorized has been studied by Berežný and Lacko [3, 4] and Grinèová, Kravecová, and Kulàè [15]. Punnen and Nair [26] studied LBOP with an additional linear constraint. Punnen and Aneja [25] and Turner et al. [32] studied the lexicographic version of LBOP. To the best of our knowledge, QBOP has not been studied in literature so far.

Most of the applications of LBOP discussed in literature translate into applications of QBOP by interpreting c_{ij} as the pairwise interaction weight of elements i and j in E . To illustrate this, let us consider the following variation of the travel agency example of Martello et al. [21]. A North American travel agency is planning to prepare a European tour package. Its clients travel from New York to London by a chartered flight. The clients have the option to choose a maximum of two tourist locations from an available set S of locations. If a client chooses locations i and j , then c_{ij} is the total tour time. There are n potential locations and the company wishes to choose $k = |S|$ locations to be included in the package so that the duration of tours for any pair of locations is approximately the same. This way, one can avoid waiting time of clients in London, after their tour and the whole group can return by the same chartered flight. The objective of the tour company can be represented as Minimizing $\max\{c_{ij} : (i, j) \in S \times S\} - \min\{c_{ij} : (i, j) \in S \times S\}$ while satisfying appropriate constraints. Another application of the model arises in balanced portfolio selection for managing investment accounts where risk estimates on pair of investment opportunities are to be considered because of hedging positions. Selection of gift items that are to be packaged into pairs of items is yet another application of the model.

In this paper we study QBOP and propose several general purpose algorithms. The polynomial solvability of these algorithms are closely related to that of an associated feasibility problem. QBOP is observed to be NP-hard even if the family \mathcal{F} of feasible solutions have very simple structure. We also investigate QBOP when the cost matrix C has a decomposable structure, i.e., $c_{ij} = a_i + b_j$ or $c_{ij} = a_i b_j$. In each of these special cases, we show that QBOP can be solved in polynomial time whenever the corresponding LBOP can be solved in polynomial time. As a consequence, we have $O(m^2 \log n)$ and $O(n^6)$ algorithms for QBOP when \mathcal{F} is chosen as spanning trees of a graph on n nodes and m edges or perfect matchings on a $K_{n,n}$, respectively. Our general purpose exact algorithms can be modified into heuristic algorithms. Some sufficient conditions are derived to speed up these algorithms and their effect is analyzed using extensive experimentation in the context of quadratic balanced knapsack problems. We also compared the heuristic solutions with exact solutions and the results establish the efficiency of our heuristic algorithms.

The paper is organized as follows. In section 2 we discuss the complexity of the problem

and introduce notations and definitions. Section 3 deals with exact and heuristic algorithms. In section 4 we present our we present polynomially solvable special cases. In section 5 we discuss the special case of the quadratic balanced knapsack problem. Experimental results are presented in section 6 followed by concluding remarks in section 7.

2 Complexity and notations

Without loss of generality, we assume that $c_{ij} \geq 0$ for otherwise we can add a large constant to all c_{ij} values to get an equivalent problem with non-negative cost values. It may be noted that when $c_{ij} \geq 0$ for all $(i, j) \in E \times E$ and $c_{ii} = 0$ for $i \in E$, QBOP reduces to the *quadratic bottleneck problem* (QBP) [7, 27]. QBP is NP-hard even if \mathcal{F} is the collection of all subsets of E with cardinality no more than k for a given k , which depends on m [27]. In fact, for such a problem, computing an ϵ -optimal solution is also NP-hard for any $\epsilon > 0$ even if $c_{ij} \in \{0, 1\}$ and $c_{ii} = 0$ [27]. As an immediate consequence, it can be verified that for the corresponding instance of QBOP, computing an ϵ -optimal solution is NP-hard for any $\epsilon > 0$. In contrast, the corresponding LBOP is polynomially solvable. Thus, the complexity of QBOP and LBOP are very different and QBOP apparently is a more difficult problem.

For a given cost matrix C and $S \in \mathcal{F}$, we denote

$$\begin{aligned} Z_{\max}(C, S) &= \max\{c_{ij} : (i, j) \in S \times S\}, \\ Z_{\min}(C, S) &= \min\{c_{ij} : (i, j) \in S \times S\} \text{ and} \\ Z(C, S) &= Z_{\max}(C, S) - Z_{\min}(C, S). \end{aligned}$$

For a given family of feasible solutions, we use the notation QBOP(C) to indicate that the cost matrix under consideration for QBOP is C . Thus, QBOP(C) and QBOP(C^*), where $C \neq C^*$, are two instances of QBOP with the same family of feasible solutions but different cost matrices C and C^* respectively.

For any two real numbers α and β such that $\alpha \leq \beta$ and cost matrix C , let $F(C, \alpha, \beta) = \{S \in \mathcal{F} : Z_{\min}(C, S) \geq \alpha \text{ and } Z_{\max}(C, S) \leq \beta\}$ and $E(C, \alpha, \beta) = \{(i, j) : c_{ij} < \alpha \text{ or } c_{ij} > \beta\}$. Then the *quadratic feasibility problem* can be stated as follows: “Given two real numbers α and β , where $\alpha \leq \beta$, test if $F(C, \alpha, \beta) \neq \emptyset$ and produce an $S \in F(C, \alpha, \beta)$ whenever $F(C, \alpha, \beta) \neq \emptyset$.”

Any solution $S \in \mathcal{F}$ can be represented by its incidence vector $x = (x_1, x_2, \dots, x_m)$, where

$$x_i = \begin{cases} 1 & \text{if } i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

The solution represented by an incidence vector x is denoted by $S(x)$. Let $\mathcal{F}_{\text{hull}}$ be the convex hull of the incidence vectors of all the elements of \mathcal{F} . Consider the cost matrix C'

given by

$$c'_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E(C, \alpha, \beta), \\ 0 & \text{otherwise.} \end{cases}$$

Then the quadratic feasibility problem has a ‘yes’ answer if and only if the optimal objective function value of the *quadratic programming problem* (QPP)

$$\begin{aligned} & \text{Minimize} \quad \sum_{i=1}^m \sum_{j=1}^m c'_{ij} x_i x_j \\ & \text{subject to} \quad x \in \mathcal{F}_{\text{hull}} \end{aligned}$$

is zero and if x^0 is the corresponding optimal basic feasible solution then $S(x^0) \in F(C, \alpha, \beta)$. Thus, the quadratic feasibility problem can be solved by solving the QPP.

The quadratic feasibility problem can also be viewed as the feasibility version of the *linear combinatorial optimization problem with conflict pairs* (LCOP), where the associated set of conflict pairs is precisely $E(C, \alpha, \beta)$; i.e, the quadratic feasibility problem has a “yes” answer if and only if the set

$$\{x : x \in \mathcal{F}_{\text{hull}}, x_i + x_j \leq 1 \text{ for } (i, j) \in E(C, \alpha, \beta)\} \quad (1)$$

is non-empty. For details on the LCOP we refer to [11, 35]. The quadratic feasibility problem discussed above is closely related to the quadratic feasibility problem studied by Punnen and Zhang [27] in the context of quadratic bottleneck problems.

3 Exact and heuristic algorithms for QBOP

Let us now consider some general results which are used in the subsequent sections to design algorithms for QBOP. Let $\mathcal{F}^* = \{S_1, S_2, \dots, S_r\}$ be a subset of \mathcal{F} satisfying the following properties:

(P1) There exists an $S_i \in \mathcal{F}^*$ which is an optimal solution to QBOP,

(P2) $Z_{\max}(C, S_1) < Z_{\max}(C, S_2) < \dots < Z_{\max}(C, S_r)$.

For any index k , $1 \leq k \leq r$, let $\pi(k)$ be the index such that $Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)}) = \min\{Z_{\max}(C, S_i) - Z_{\min}(C, S_i) : 1 \leq i \leq k\}$. Then, clearly, $S_{\pi(r)}$ is an optimal solution to QBOP. Let Ω be a real number such that $\Omega \geq Z_{\min}(C, S_i)$ for any optimal solution S_i for QBOP in \mathcal{F}^* .

Theorem 1. *For any $1 \leq k \leq r$, if $Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)}) + \Omega \leq Z_{\max}(C, S_k)$ then $S_{\pi(k)}$ is an optimal solution to QBOP.*

Proof. Suppose $S_{\pi(k)}$ is not an optimal solution to QBOP. Then there is an optimal solution S_i to QBOP in \mathcal{F}^* such that $i > k$. Thus, $Z_{\max}(C, S_i) - Z_{\min}(C, S_i) < Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)})$. Then

$$\begin{aligned} Z_{\max}(C, S_i) &< Z_{\min}(C, S_i) + Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)}) \\ &\leq \Omega + Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)}) \\ &\leq Z_{\max}(C, S_k). \end{aligned}$$

Thus, by (P2), $i \leq k$, a contradiction. \square

Theorem 2. *If $S_{\pi(k)}$ is not an optimal solution to QBOP then there exists an optimal solution $S_q \in \mathcal{F}^*$ such that $q > k$ and $Z_{\min}(C, S_q) > Z_{\max}(C, S_k) - Z_{\max}(C, S_{\pi(k)}) + Z_{\min}(C, S_{\pi(k)})$*

Proof. Since $S_{\pi(k)}$ is not optimal, by property (P1) there exists an optimal solution S_q in \mathcal{F}^* such that $q > k$. Suppose $Z_{\min}(C, S_q) \leq Z_{\max}(C, S_k) - Z_{\max}(C, S_{\pi(k)}) + Z_{\min}(C, S_{\pi(k)})$. Then $Z_{\min}(C, S_q) \leq Z_{\max}(C, S_q) - Z_{\max}(C, S_{\pi(k)}) + Z_{\min}(C, S_{\pi(k)})$. Thus, $Z_{\max}(C, S_q) - Z_{\min}(C, S_q) \geq Z_{\max}(C, S_{\pi(k)}) - Z_{\min}(C, S_{\pi(k)})$ establishing that $S_{\pi(k)}$ is also an optimal solution to QBOP, a contradiction. \square

Theorems 1 and 2 assist us in improving the average performance of our algorithms. Corresponding results can be obtained by considering solutions that satisfy another set of properties. Suppose $\mathcal{F}^0 = \{S_1, S_2, \dots, S_h\}$ be a subset of \mathcal{F} satisfying the following properties.

(P3) There exists an $S_i \in \mathcal{F}^0$ which is an optimal solution to QBOP

(P4) $Z_{\min}(C, S_1) > Z_{\min}(C, S_2) > \dots > Z_{\min}(C, S_h)$.

Choose an index $\sigma(k)$ such that $Z_{\max}(C, S_{\sigma(k)}) - Z_{\min}(C, S_{\sigma(k)}) = \min\{Z_{\max}(C, S_i) - Z_{\min}(C, S_i) : 1 \leq i \leq k\}$. Then $S_{\sigma(k)}$ is an optimal solution to QBOP. Let Δ be a real number such that $\Delta \leq Z_{\max}(C, S_i)$ for any optimal solution S_i for QBOP in \mathcal{F}^0 .

Theorem 3. *For any $1 \leq k \leq h$, if $\Delta - Z_{\max}(C, S_{\sigma(k)}) + Z_{\min}(C, S_{\sigma(k)}) \geq Z_{\min}(C, S_k)$ then $S_{\sigma(k)}$ is an optimal solution to QBOP.*

Theorem 4. *If $S_{\sigma(k)}$ is not an optimal solution to QBOP then there exists an optimal solution $S_d \in \mathcal{F}^0$ such that $d > k$ and $Z_{\max}(C, S_d) < Z_{\max}(C, S_{\sigma(k)}) - Z_{\min}(C, S_{\sigma(k)}) + Z_{\min}(C, S_k)$*

The proofs of Theorems 3 and 4 can be obtained along the same lines as that of Theorems 1 and 2 and hence omitted.

Conditions similar to Theorems 1 to 4 have been used by many authors in the context of different optimization problems involving linear terms [1, 22, 19, 24, 26]. The effect of such conditions are not tested in the context of quadratic type problems. One of the goals of our experimental analysis was to test the efficacy of Theorems 1 to 4 in the development of practical algorithms.

3.1 The double threshold algorithm

The basic idea of this algorithm is similar to that used by Martello et al. [21] for solving LBOP. The difference, however, is that we are using the quadratic feasibility problem discussed in the last section instead of a simple feasibility test considered in [21] for LBOP. This is a significant deviation as it alters the problem complexity substantially. The validity proof of our algorithm follows along the same line as that of the double threshold algorithm for LBOP discussed in [21]. We also use the conditions provided in Theorems 1 and 2 to enhance our search for an optimal solution.

Let $w_1 < w_2 < \dots < w_p$ be an ascending arrangement of distinct elements of the cost matrix C and $w_{p+1} = \infty$. These w_i values are the candidates for $Z_{\max}(C, S)$ and $Z_{\min}(C, S)$ for any feasible solution S . The algorithm performs a bottom-up sequential search by maintaining a lower threshold L and an upper threshold U , and tests if $F(C, L, U) \neq \emptyset$. If the answer is ‘yes’, the lower threshold is increased and if the answer is ‘no’, the upper threshold is increased. The lower and upper threshold values are chosen amongst $\{w_1, w_2, \dots, w_p\}$. At any stage of the algorithm, if a feasible solution is obtained with the QBOP objective function value as zero, the algorithm is terminated since we have an optimal solution. Let $L = w_\ell$ and $U = w_u$ for some ℓ and u , $\ell \leq u$. If $F(C, w_\ell, w_u) = \emptyset$ then U is increased to w_{u+1} . Otherwise, we choose an $S \in F(C, w_\ell, w_u)$ and L can be increased to w_{v+1} , where $w_v = \min\{c_{ij} : (i, j) \in S \times S\}$, and the best solution identified so far is updated, if necessary. Note that $w_v \geq w_\ell$.

We also try to exploit the conditions of Theorems 1 and 2 for early detection of an optimal solution or rapid increase in the lower threshold (and hence, possibly the upper threshold). Let $t \leq p$ be the total number of times L is updated and let $\mathcal{F}^* = \{S_1, S_2, \dots, S_t\}$ be the set of solutions generated. The indexes are selected such that S_i is generated before S_{i+1} . Then \mathcal{F}^* satisfies the properties (P1) and (P2). Thus, the sufficient condition of Theorem 1 can be used to detect optimality in any iteration, whenever the condition is satisfied. If it is satisfied then the best solution identified so far is indeed optimal and the algorithm terminates. Otherwise, we try to increase the lower threshold L (and hence possibly the upper threshold U) rapidly using the conditions of Theorem 2. If the algorithm is not terminated using any of the conditions discussed above, then the search completes when U or L becomes ∞ and the best solution produced during the search is selected as the output which is an optimal solution to QBOP. A formal description of the bottom-up double threshold algorithm (BDT algorithm) is given in Algorithm 1.

With the assumption that the dominating complexity of this algorithm in each iteration is the complexity of testing the condition if $F(C, w_l, w_u) \neq \emptyset$ or not. If this test can be performed in $O(\phi_1(m))$ time, then the BDT-algorithm terminates in $O(m\phi_1(m))$ time. For most problems of practical interest, testing if $F(C, w_l, w_u) \neq \emptyset$ or not is NP-hard. By performing this test using heuristic algorithms, we can get a heuristic algorithm to solve QBOP. The computational issues associated with this approach are discussed in detail in the section of experimental analysis of algorithms.

Just like the BDT algorithm, it is possible to obtain another double threshold algorithm using a top-down search. In this case, we start with the upper and lower threshold values at

Algorithm 1: The BDT Algorithm

```
1 Let  $w_1 < w_2 < \dots < w_p$  be an ascending arrangement of distinct values of  $e_{ij}$  for  
    $(i, j) \in E \times E$ ;  
2  $l \leftarrow 1$ ;  $u \leftarrow 1$ ;  $sol \leftarrow \emptyset$ ;  $obj \leftarrow \infty$ ;  
3 Compute the parameter  $\Omega$ ;  
4 while  $l \leq p$  and  $u \leq p$  do  
5   if  $F(C, w_l, w_u) \neq \emptyset$  then  
6     Choose an  $S \in F(C, w_l, w_u)$ ;  
7     if  $Z(C, S) < obj$  then  $obj \leftarrow Z(C, S)$ ;  $sol \leftarrow S$ ;  
8     if  $obj = 0$  or  $obj + \Omega \leq w_t$  then return  $sol$ ;  
9     Choose smallest  $k$  such that  $w_k > \max\{Z_{\min}(C, S), Z_{\max}(C, S) - obj\}$ ;  
10     $l \leftarrow k$ ;  
11    if  $w_l > w_u$  then  $u \leftarrow k$ ;  
12  else  
13     $u \leftarrow u + 1$   
14  end if  
15 end while  
16 return  $obj$  and  $sol$ 
```

w_p and systematically decrease the threshold values. The algorithm makes use of Theorems 3 and 4 to improve the search process, along similar lines as in the BDT algorithm where the threshold values are systematically increased. The resulting algorithm is called top-down double threshold algorithm (TDT algorithm). The detailed description of various steps of this algorithm can be easily constructed in view of the BDT algorithm and, therefore, omitted.

3.2 Iterative bottleneck algorithms

Let us now discuss two additional algorithms for solving QBOP which solve a sequence of quadratic bottleneck problems. The worst case complexities of these algorithms, in general, are higher than that of the BDT-algorithm, but their average performance is expected to be better. A quadratic bottleneck problem of type 1 (QBP1) is defined as

$$\begin{aligned} \text{QBP1:} \quad & \text{Minimize } Z_{\max}(C, S) \\ & \text{subject to } S \in \mathcal{F}. \end{aligned}$$

We denote an instance of QBP1 with cost matrix C as $\text{QBP1}(C)$. The problem QBP1 was investigated by Burkard [7] and Punnen and Zhang [27].

To develop our iterative bottleneck algorithms, we consider a generalization of QBOP, where a restriction on the upper threshold is imposed on the feasible solutions. Consider the problem

$$\begin{aligned} \text{QBOP1}(C, \alpha): \quad & \text{Minimize } Z_{\max}(C, S) - Z_{\min}(C, S) \\ & \text{subject to } S \in \mathcal{F}, \\ & Z_{\min}(C, S) \geq \alpha \end{aligned}$$

When $\alpha = \min\{c_{ij} : (i, j) \in E \times E\}$ QBOP1(C, α) reduces to QBOP(C). Let C' be an $m \times m$ matrix defined by

$$c'_{ij} = \begin{cases} M & \text{if } c_{ij} < \alpha, \\ c_{ij} & \text{otherwise,} \end{cases}$$

where M is a large number.

Theorem 5. *Let S^0 be an optimal solution to QBP1 with cost matrix C' and q be the index such that $w_q = Z_{\min}(C, S^0)$.*

- (1) *If $Z_{\max}(C, S^0) = M$ then QBOP1(C, α) is infeasible.*
- (2) *If $Z_{\max}(C, S^0) < M$ and $Z_{\max}(C, S^0) = Z_{\min}(C, S^0)$ then S^0 is an optimal solution to QBOP1(C, α).*
- (3) *If conditions (1) and (2) above are not satisfied, then either S^0 is an optimal solution to QBOP1(C, α) or an optimal solution to QBOP1(C', γ) is also optimal to QBOP1(C, α), where $\gamma = w_{q+1}$.*

Proof. The proof of (1) and (2) are straightforward. Let us now prove (3). Let $\chi = \{S \in \mathcal{F} : \alpha \leq Z_{\min}(C, S) \leq Z_{\min}(C, S^0)\}$. By definition of χ

$$Z_{\min}(C, S^0) \geq Z_{\min}(C, S) \text{ for all } S \in \chi. \quad (2)$$

Since condition (1) of the theorem is not satisfied, by optimality of S^0 to QBP1 with cost matrix C' , we have

$$Z_{\max}(C, S^0) \leq Z_{\max}(C, S) \text{ for all } S \in \chi. \quad (3)$$

Multiply inequality (2) by -1 and adding to inequality (3) we have $Z(C, S^0) \leq Z(C, S)$ for all $S \in \chi$. Thus, either S^0 is an optimal solution to QBOP1(C, α) or there exists an optimal solution S to QBOP1(C, α) satisfying $Z_{\min}(C, S) > Z_{\min}(C, S^0)$ and the result follows. \square

In view of Theorem 5, we can solve QBOP as a sequence of QBP1 problems. In each iteration, the algorithm maintains a lower threshold α and constructs a modified cost matrix C' which depends on the value of α . Then, using an optimal solution to QBP1 with cost matrix C' , the lower threshold is systematically increased until infeasibility with respect to the threshold values is reached or optimality of one of the solutions generated so far is identified using condition (2) of Theorem 5. Let $F^1 = \{S_{\rho_1}, S_{\rho_2}, \dots, S_{\rho_t}\}$ be the set of solutions generated for various QBP1 problems, where S_{ρ_i} is generated before $S_{\rho_{i+1}}$, $1 \leq i \leq t-1$. Then by choosing $F^* = F^1$, these solutions satisfy properties (P1) and (P2). Thus, Theorem 1 can be used to detect optimality early and Theorem 2 may be used to increase the lower threshold rapidly. If the algorithm is not terminated using an optimality condition, it compares all the solutions generated by the QBP1 solver and outputs the overall best solution with respect to the QBOP objective function. The resulting algorithm is called the

type 1 iterative bottleneck algorithm (IB1 algorithm) and its formal description is given in Algorithm 2.

Algorithm 2: The IB1 Algorithm

```

1 Let  $w_1 < w_2 < \dots < w_p$  be an ascending arrangement of all distinct values of
    $\{c_{ij} : (i, j) \in E \times E\}$ ;
2  $C' \leftarrow C$ ;  $obj \leftarrow \infty$ ;  $sol \leftarrow \emptyset$ ,  $M \leftarrow 1 + w_p$ ;  $z_0 = w_p$ ;
3 Compute the parameter  $\Omega$ ;
4 while  $z_0 \neq M$  do
5   Solve QBP1( $C'$ );
6   if If QBP1( $C'$ ) is infeasible then return  $\emptyset$  and  $\infty$ ;
7   else let  $S$  be the solution of QBP1( $C'$ );
8    $z_0 \leftarrow \max\{c'_{ij} : (i, j) \in S \times S\}$ ;
9   if  $z_0 < M$  then
10    if  $Z(C, S) < obj$  then  $sol \leftarrow S$ ;  $obj \leftarrow Z(C, S)$ ;
11    if  $obj = 0$  or  $obj + \Omega \leq w_{\tau_1}$  then return  $obj$  and  $sol$ ;
12     $L \leftarrow \max\{Z_{\min}(C, S), Z_{\max}(C, S) - obj\}$ ;
13     $c'_{ij} \leftarrow \begin{cases} c_{ij} & \text{if } c_{ij} > L, \\ M & \text{otherwise} \end{cases}$  for each  $i, j \in E$ ;
14  end if
15 end while
16 return  $obj$  and  $sol$ 

```

The IB1 algorithm solves at most m problems of the type QBP1. Thus, if QBP1 can be solved in $O(\phi_2(m))$ time, then QBOP can be solved in $O(m\phi_2(m))$ time. By solving QBP1 using a heuristic, we get a heuristic version of the IB1 algorithm.

QBOP can also be solved as a sequence of quadratic bottleneck problems of the maxmin type, which we call a *quadratic bottleneck problem of type 2* (QBP2). Formally, QBP2 can be stated as follows:

QBP2: Maximize $Z_{\min}(C, S)$
 subject to $S \in \mathcal{F}$.

QBP2 can be reformulated as QBP1 or the algorithms for QBP1 [27] can be modified to solve QBP2 directly.

Now, for any real number β , consider the problem:

QBOP2(C, β): Minimize $Z_{\max}(C, S) - Z_{\min}(C, S)$
 subject to $S \in \mathcal{F}$,
 $Z_{\max}(C, S) \leq \beta$

When $\beta = \max\{c_{ij} : (i, j) \in E \times E\}$, QBOP2(C, β) reduces to QBOP(C). Define the cost

matrix \tilde{C} defined by

$$\tilde{c}_{ij} = \begin{cases} -M & \text{if } c_{ij} > \beta \text{ or } c_{ij} < \alpha \\ c_{ij} & \text{otherwise.} \end{cases}$$

where M is a large number.

Theorem 6. *Let S^0 be an optimal solution to QBP2 with cost matrix \tilde{C} and r be the index such that $w_r = Z_{\max}(C, S^0)$.*

1. *If $Z_{\min}(\tilde{C}, S^0) = -M$ then QBOP2(C, β) is infeasible.*
2. *If $Z_{\min}(\tilde{C}, S^0) > -M$ and $Z_{\max}(C, S^0) = Z_{\min}(C, S^0)$ then S^0 is an optimal solution to QBOP2(C, β).*
3. *If conditions (1) and (2) are not satisfied, then either S^0 is an optimal solution to QBOP2(C, β) or an optimal solution to QBOP2(\tilde{C}, γ) is also optimal to QBOP2(C, β) where $\gamma = w_{r+1}$.*

The proof of this theorem can be constructed by appropriate modifications in the proof of Theorem 5 and hence is omitted.

In view of Theorem 6, we can solve QBOP as a sequence of QBP2 problems. In each iteration, the algorithm maintains a lower threshold α and an upper threshold β and construct a modified cost matrix \tilde{C} which depends on the value of α and β . Using an optimal solution to QBP2 with cost matrix \tilde{C} , the upper threshold is systematically decreased and the process is continued until infeasibility with respect to the threshold values is reached or optimality of one of the solutions generated so far is identified using condition (2) of Theorem 6. Let $F^2 = \{S_{\eta_1}, S_{\eta_2}, \dots, S_{\eta_t}\}$ be the set of solutions generated for various QBP2 problems. The indexes are selected such that S_{η_i} is generated before $S_{\eta_{i+1}}$, $1 \leq i \leq t-1$. Then by choosing $F^0 = F^2$, these solutions satisfy properties (P3) and (P4). Thus, Theorem 3 may be used to detect optimality early in some cases and Theorem 4 may be used to decrease the upper threshold rapidly. If the algorithm is not terminated using an optimality condition, it compares the solutions generated by the QBP2 solver and outputs the overall best solution with respect to the QBOP objective function. The resulting algorithm is called the *type 2 iterative bottleneck algorithm* (IB2-algorithm). A formal description of the IB2 algorithm is omitted as it can be obtained by appropriate modifications of the IB1 algorithm.

The IB2 algorithm solves at most m problems of the type QBP2. Thus, if QBP2 can also be solved in $O(\phi_3(m))$ time then QBOP can be solved in $O(m\phi_3(m))$ time. By solving QBP2 using a heuristic, we get a heuristic version of the IB2 algorithm.

3.3 The double bottleneck algorithm

Note that Algorithm IB1 sequentially increases the lower threshold value while the algorithm IB2 sequentially decreases the upper threshold value. The two algorithms can be combined to generate another algorithm that alternately increases the lower threshold and decreases the upper threshold. The operations of increasing the lower threshold or decreasing the upper threshold are carried out by solving a BQP1 and QBP2, respectively. The resulting

algorithm is called the *double bottleneck algorithm* (DB-Algorithm). The validity of the DB-algorithm follows from Theorems 5 and 6 and the validity of algorithms IB1 and IB2. A formal description of the DB Algorithm is given in Algorithm 3.

Algorithm 3: The DB Algorithm

```

1  Let  $w_1 < w_2 < \dots < w_p$  be an ascending arrangement of all distinct values of
    $\{c_{ij} : (i, j) \in E \times E\}$ ;
2   $C' \leftarrow C$ ;  $obj \leftarrow \infty$ ;  $sol \leftarrow \emptyset$ ,  $M \leftarrow 1 + w_p$ ;  $\bar{z} = w_p$ ;  $\tilde{z} = w_1$ ;
3  while  $\bar{z} \neq M$  and  $\tilde{z} \neq -M$  do
4    Solve QBP1( $C'$ );
5    if If QBP1( $C'$ ) is infeasible then return  $\emptyset$  and  $\infty$ ;
6    else let  $S$  be the solution of QBP1( $C'$ );
7     $\bar{z} \leftarrow \max\{c'_{ij} : (i, j) \in S \times S\}$ ;
8    if  $z_0 < M$  then
9      if  $Z(C, S) < obj$  then  $sol \leftarrow S$ ;  $obj \leftarrow Z(C, S)$ ;
10     if  $obj = 0$  then return  $sol$  and  $obj$ ;
11      $c'_{ij} \leftarrow \begin{cases} c_{ij} & \text{if } c_{ij} > Z_{\min}(C, S), \\ M & \text{otherwise} \end{cases}$  for each  $i, j \in E$ ;
12   end if
13   Solve QBP2( $C'$ ); let  $S$  be the resulting solution;
14    $\tilde{z} \leftarrow \min\{c'_{ij} : (i, j) \in S \times S\}$ ;
15   if  $\tilde{z} > -M$  then
16     if  $Z(C, S) < obj$  then  $sol \leftarrow S$ ;  $obj \leftarrow Z(C, S)$ ;
17     if  $obj = 0$  then return  $sol$  and  $obj$ ;
18      $c'_{ij} \leftarrow \begin{cases} c_{ij} & \text{if } c_{ij} < Z_{\max}(C, S) \\ -M & \text{otherwise} \end{cases}$  for each  $i, j \in E$ ;
19   end if
20 end while
21 return  $opt$  and  $sol$ 

```

4 Polynomially solvable cases

Let us now consider some special cases of QBOP that can be solved in polynomial time. We first consider the *decomposable cost matrix*, where $c_{ij} = a_i + b_j$ for given $a_i \geq 0$, $b_j \geq 0$ and $i, j \in E$. We denote such an instance of a QBOP by QBOP($a + b$). Let $Z^+(C, S)$ denote the

objective function of QBOP($a + b$). Then,

$$\begin{aligned}
Z^+(C, S) &= \max\{c_{ij} : (i, j) \in S \times S\} - \min\{c_{ij} : (i, j) \in S \times S\} \\
&= \max\{a_i + b_j : (i, j) \in S \times S\} - \min\{a_i + b_j : (i, j) \in S \times S\} \\
&= \max\{a_i : i \in S\} + \max\{b_i : i \in S\} - \min\{a_i : i \in S\} - \min\{b_i : i \in S\} \quad (4) \\
&= \max\{a_i : i \in S\} + \max\{-a_i : i \in S\} + \max\{b_i : i \in S\} - \min\{b_i : i \in S\} \quad (5)
\end{aligned}$$

Let w_i be some prescribed weight of $i \in E$ and $g : \mathcal{F} \rightarrow \mathbb{R}$. Duin and Volgenant [12] showed that combinatorial optimization problems of the type

$$\begin{aligned}
\text{COP}(g): \quad & \text{Minimize } \max\{w_i : i \in S\} + g(S) \\
& \text{subject to } S \in \mathcal{F}.
\end{aligned}$$

can be solved in $O(m\zeta(m))$, where $\zeta(m)$ is the complexity of minimizing $g(S)$ over \mathcal{F} . Note that

$$Z^+(C, S) = \max\{a_i : i \in S\} + g(S),$$

where $g(S) = \max\{-a_i : i \in S\} + g_1(S)$ and $g_1(S) = \max\{b_i : i \in S\} - \min\{b_i : i \in S\}$. But minimizing $g_1(S)$ over \mathcal{F} is precisely the LBOP [21]. Thus, recursively applying the results of Duin and Volgenant [12], QBOP($a + b$) can be solved in $O(m^2\eta(m))$ time, where $\eta(m)$ is the complexity of an LBOP with the same family of feasible solutions as that of the QBOP($a + b$).

The complexity of this algorithm can be improved by careful organization of the computations. Let α and β be two real numbers such that $\alpha \leq \beta$. Let $Q_{\alpha, \beta} = \{i : \alpha \leq a_i \leq \beta\}$. Let $\gamma = \min\{b_i : i \in Q_{\alpha, \beta}\}$ and $\delta = \max\{b_i : i \in Q_{\alpha, \beta}\}$. Consider the constrained LBOP:

$$\begin{aligned}
\text{LBOP}(\alpha, \beta): \quad & \text{Minimize } \max\{b_i : i \in S\} - \min\{b_i : i \in S\} \\
& \text{subject to } s \in \mathcal{F}, \\
& \max\{b_i : i \in S\} \leq \delta, \\
& \min\{b_i : i \in S\} \geq \gamma.
\end{aligned}$$

LBOP(α, β) can be solved using the double threshold algorithm for LBOP with very simple modification by choosing the starting and ending threshold values as γ and δ , respectively, and the complexity of the algorithm for LBOP in [21] is unaltered when adapted to solve LBOP(α, β). Now we can solve QBOP($a+b$) as a sequence of problems of the type LBOP(α, β). As in [21], we use a double threshold search using the a_i values and simply replace the feasibility problem in [21] by the problem LBOP(α, β). Thus, if LBOP(α, β) can be solved in $O(\eta(m))$ time, then QBOP($a + b$) can be solved in $O(m\eta(m))$.

Note that when feasible solutions are spanning trees of a graph on n nodes and m edges, LBOP(α, β) can be solved in $O(m \log n)$ time [14] and, hence, the resulting QBOP($a + b$) can be solved in $O(m^2 \log n)$ time. When \mathcal{F} is the family of perfect matchings in a complete bipartite graph $K_{n,n}$ then LBOP(α, β) can be solved in $O(n^4)$ time [21] and, hence, QBOP($a + b$) can be solved in $O(n^6)$ time.

Let us now consider the case when $c_{ij} = a_i b_j$ for $(i, j) \in E \times E$ where $a_i \geq 0, b_i \geq 0$ for $i = 1, 2, \dots, n$. The corresponding instance of QBOP is denoted by QBOP(ab). Now, for any feasible solution $S \in \mathcal{F}$,

$$\begin{aligned} Z^*(C, S) &= \max\{c_{ij} : (i, j) \in S \times S\} - \min\{c_{ij} : (i, j) \in S \times S\} \\ &= \max\{a_i b_j : (i, j) \in S \times S\} - \min\{a_i b_j : (i, j) \in S \times S\} \\ &= \max\{a_i : i \in S\} \max\{b_i : i \in S\} - \min\{a_i : i \in S\} \min\{b_i : i \in S\}. \end{aligned} \quad (6)$$

By fixing $\max\{a_i : i \in S\}$ and $\min\{a_i : i \in S\}$ between a lower threshold α and an upper threshold β , we can solve QBOP(ab) similar to QBOP($a + b$) as a sequence of problems of the type LBOP(α, β) and the only difference is that we use the objective function $Z^*(C, S)$ instead of $Z^+(C, S)$ when comparing various feasible solutions generated in the course of the algorithm. The complexity of the algorithm will be the same as that for the problem QBOP($a+b$) we discussed earlier.

5 The quadratic balanced knapsack problem

Let us now consider a specific case of QBOP called the *quadratic balanced knapsack problem* (QBalkP) which can be defined as follows:

$$\begin{aligned} &\text{Minimize } \max\{c_{ij} : (i, j) \in E \times E, x_i = x_j = 1\} - \min\{c_{ij} : (i, j) \in E \times E, x_i = x_j = 1\} \\ &\text{subject to } \sum_{j=1}^m a_j x_j \geq b \\ &x_j \in \{0, 1\} \text{ for } j \in E. \end{aligned} \quad (7)$$

By choosing $\mathcal{F} \subseteq 2^E$ such that $S \in \mathcal{F}$ implies $\sum_{i \in S} a_i \geq b$ we can see that QBalkP is an instance QBOP. The compact representation of \mathcal{F} is given by the constraints (7) and (8). QBalkP can be used to model the travel agency example and portfolio selection examples discussed in section 1. Since the quadratic bottleneck knapsack problem (QBotKP) [36] is a special case of QBalkP and QBotKP is strongly NP-hard, QBalkP is also strongly NP-hard. (Note that the LBOP version of QBalkP is solvable in polynomial time.) QBalkP can be formulated as a mixed integer program:

$$\begin{aligned} &\text{Minimize } u - v \\ &\text{Subject to } \sum_{j=1}^m a_j x_j \geq b, \\ &u \geq c_{ij} y_{ij} \text{ for } (i, j) \in E \times E, c_{ij} \neq 0, \\ &v \leq c_{ij} y_{ij} + M(1 - y_{ij}) \text{ for } (i, j) \in E \times E, c_{ij} \neq M, \\ &y_{ij} - x_i \leq 0 \text{ for } (i, j) \in E \times E, \\ &y_{ij} - x_j \leq 0 \text{ for } (i, j) \in E \times E, \\ &x_i + x_j - y_{ij} \leq 1 \text{ for } (i, j) \in E \times E, \\ &x_j \in \{0, 1\}, \text{ for } j \in E, \\ &0 \leq u, v \leq M, \end{aligned}$$

where $M = \max\{c_{ij} : 1 \leq i, j \leq n\}$. Solving the mixed integer programming formulation given above becomes difficult as the problem size increases. However, we can use the general purpose algorithms developed in the previous section to solve QBalKP. To use the algorithms IB1, IB2, and DB, we can make use of the algorithm of Zhang and Punnen [36] as the QBP1 (QBP2) solver. To apply the BDT and TDT algorithms, we need an algorithm to solve the corresponding quadratic feasibility problem.

Recall that the *quadratic feasibility problem* is: “Given two real numbers α and β , where $\alpha \leq \beta$, test if $F(C, \alpha, \beta) \neq \emptyset$ and produce an $S \in F(C, \alpha, \beta)$ whenever $F(C, \alpha, \beta) \neq \emptyset$.”

In section 2, we indicated that a quadratic feasibility problem can be solved as a combinatorial optimization problem with conflict pair constraints [11, 35]. We now observe that the quadratic feasibility problem for QBKP can be solved by solving the maximum weight independent set problem (MWIP) on a graph with node set E and edge set $E(\alpha, \beta)$. An integer programming representation of this MWIP is given below.

$$\begin{aligned} \text{MWIP:} \quad & \text{Maximize } \sum_{i=1}^m a_i x_i \\ & \text{subject to } x_i + x_j \leq 1 \text{ for } (i, j) \in E(\alpha, \beta), \\ & \quad x_j \in \{0, 1\}, \text{ for } j \in E. \end{aligned}$$

Let $x^* = (x_1^*, x_2^*, \dots, x_m^*)$ be an optimal solution to the MWIP and z^* be its optimal objective function value. Then $F(C, \alpha, \beta) \neq \emptyset$ if and only if $z^* \geq b$. Thus, we can use an MWIP solver to implement the algorithms discussed in the previous section for the special case of QBalKP. The solution of the quadratic feasibility problem discussed above is closely related to the quadratic feasibility problem studied by Zhang and Punnen [36] for the quadratic bottleneck knapsack problem with appropriate differences to handle the QBalKP objective.

6 Computational Experiments

In this section we report results of extensive experimental analysis conducted on randomly generated QBalKP instances. The objective of the experiments is to assess the relative performance of various algorithms developed in section 3. We have implemented exact and heuristic versions of these algorithms and compared the outcomes in terms of solution quality and computational time. Our experiments also examined the effectiveness of the conditions provided by Theorems 1, 2, 3, and 4 for early detection of optimality and rapid advancement through the search intervals. All the experiments were conducted on an Intel i7-2600 CPU based PC. The algorithms are implemented in C#, and CPLEX 12.4 was used to solve the mixed integer programming problems within our implementations. x86-64 instruction set was used and concurrency was not allowed in our the algorithms as well as in CPLEX.

All the algorithms discussed in this paper (except the MIP formulation of QBalKP) require a feasibility test procedure and the dominating complexity of these algorithms in each iteration is that of this procedure. Recall that a feasibility test answers the question if there exists a feasible solution to the QBalKP and, whenever the answer is ‘yes’, it generates

such a solution. In section 5 we observed that this can be achieved by solving a maximum weight independent set problem. We can also use other variations of this approach to test feasibility and the empirical behavior of different variations could be different. Since the feasibility test is carried out several times in the algorithm, the effect of different variations of the feasibility tests could affect the the running time as well as solution quality (for heuristic algorithms). For definiteness and simplicity, we have restricted our implementation to three different feasibility test procedures which are summarized below:

FT1: Solve the maximum weight independent set problem as described in section 5 and then compare the objective value $\sum_{i=1}^m a_i x_i$ with b . If $\sum_{i=1}^m a_i x_i \geq b$, the resulting solution $S(x)$ is a feasible solution to the QBalKP. Otherwise the answer is ‘no’. We used CPLEX 12.4 to solve the resulting integer program.

FT2: Consider the maximum weight independent set problem as described in section 5. Choose the objective function coefficients to be zero¹ and add the new constraint

$$\sum_{i=1}^m a_i x_i \geq b$$

to the formulation. We used CPLEX 12.4 to solve the resulting constrained maximum weight independent set problem. The CPLEX solver stops as soon as a feasible solution is found. When using this feasibility test procedure, we set the ‘MIP Emphasis’ parameter of CPLEX to ‘Emphasize feasibility over optimality’ which in our experiments provided the best performance.

FT3: Solve the integer program defined in FT2 by providing a time limit for the mixed integer programming solver. Note that if the solver fails to find a feasible solution in the allowed time limit, there is no guarantee that a feasible solution does not exist and, thus, using such a procedure in any of the algorithms turns an exact algorithm into a heuristic. Because of this heuristic decision, the properties (P1) and (P2) may not hold precisely. Nevertheless, we make a heuristic assumption that these properties hold and proceed accordingly.

As indicated earlier, even for these special cases of feasibility tests, the solutions returned by these feasibility tests could be different and, thus, not only the execution time of each feasibility test but also the optimization process itself for a QBalKP algorithm may vary even for the exact feasibility tests FT1 and FT2.

We use the following notations to represent our algorithms under different parameter settings. MIP stands for the mixed integer programming formulation of the QBalKP solved with CPLEX (the parameters of CPLEX are default). BDT, IB and DB denote the BDT, IB1 and DB algorithms, respectively, where the effect of Theorem 1 is suppressed. By default, we use feasibility test FT2. If ‘B’ is added to the name of an algorithm (such as

¹One can use any objective function in this feasibility test. If the objective function is not a constant, it is useful to force the solver to stop after it finds the first feasible solution. However, our experiments showed that using the original objective function in this feasibility test slows down the algorithms.

BDT^B or IB^B), feasibility test FT1 is used. BDT^Ω and IB^Ω stand for the variations of BDT and IB, where early optimality detection is guaranteed by Theorem 1 is enabled. BDT^t denotes the heuristic version of the BDT algorithm, where t is the time limit prescribed for each feasibility test (FT3). IB^t and DB^t denote the heuristic version of the corresponding algorithms, where t is the time limit prescribed for each feasibility test (FT3) within the QBP1 solving procedure.

6.1 Test problems

Since this is the first time when the QBalKP is considered in the literature, we have developed a class of test instances for the problem. These test instances are random problems constructed as follows. For each test instance, we are given a triple (m, σ, s) , where $m > 1$ is an integer, $\sigma > 0$ and $0 \leq s \leq 1$. We first generate an $m \times m$ matrix $C' = (c'_{ij})$, where c'_{ij} is a normally distributed random integer with mean $\mu = 0$ and standard deviation σ as given. Then the matrix $C = (c_{ij})$ is generated, where $c_{ij} = c'_{ij} - \min_{rs} c'_{rs}$. This guarantees that $c_{ij} \geq 0$. Then, an m -vector (a_i) , where a_i a uniformly distributed random integer in the range $0 \leq a_i \leq 1000$, is generated. Finally, b is selected as a uniformly distributed random integer in the range $\lfloor 250ms \rfloor \leq b \leq \lfloor 750ms \rfloor$. Observe that $E[\sum_{i \in S} a_i] = E[b]$ if $S \subseteq E$ such that $|S| = ms$, where $E[x]$ is the expected value of x . In other words, by varying the value of s , one can control the number of non-zeros in an optimal solution to the QBalKP instance.

It may be noted that the instances we generated are symmetric in the following sense; replacing c_{ij} with $(\max_{i'j'} c_{i'j'}) - c_{ij}$ would not, on average, change the properties of the matrix C . Hence, the algorithms IB1 and IB2 are expected to show similar average performance. Thus, hereafter, we do not discuss the algorithm IB2 and denote IB1 as IB. Likewise, BDT and TDT algorithms are expected to have similar average performance. Thus, we do not consider the TDT algorithm in our experimental analysis and focus on the BDT algorithm.

Figure 1 indicates relative performance of the BDT and IB algorithms as a function of the parameter s . We set $m = 100$ and $\sigma = 100$ in this experiment.

It appears that the instance with either large or small values of s are relatively easy to solve. For small values of s , each feasibility test takes only a small amount of time since it is easy to find a feasible solution if b is small. For large values of s , each feasibility test also takes a relatively small time since many such problems become infeasible. Another interesting observation is that the number of iterations of the BDT algorithm almost does not depend on s (indeed, even if the problem is infeasible, the BDT algorithm will make p iterations) while it varies significantly for the IB algorithm making it significantly faster for certain class of instances.

Figure 2 indicates the performance of the BDT and IB algorithms as a function of the parameter σ of the instance. The value of s is set in this experiment to $s = 0.1$ and $m = 100$.

One can see that the random instances become harder with increase in the value of σ . Indeed, a larger σ leads to a larger number p of distinct weights c_{ij} which, in turn, increases the number of iterations of the algorithm. However, p is limited by m^2 and, thus, it grows slower than σ . In fact, the number of iterations of the BDT algorithm is approximately

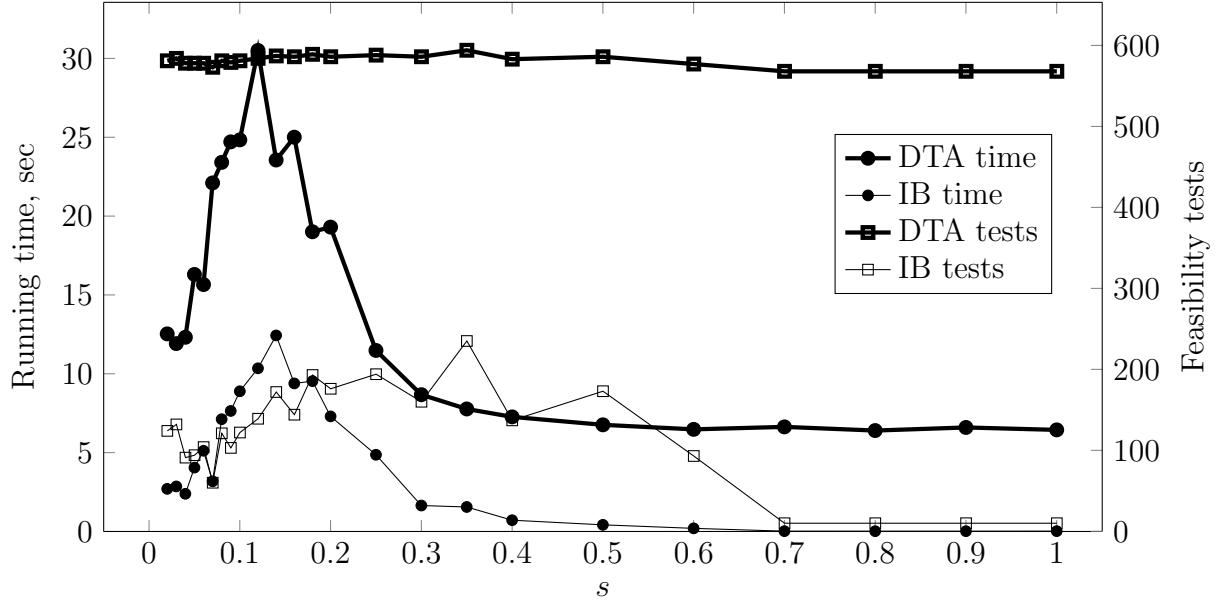


Figure 1: Analysis of the random instances with different values of parameter s . The lines with circle marks indicate the running time of the algorithms and the lines with square marks indicate the number of feasibility tests applied within the algorithms.

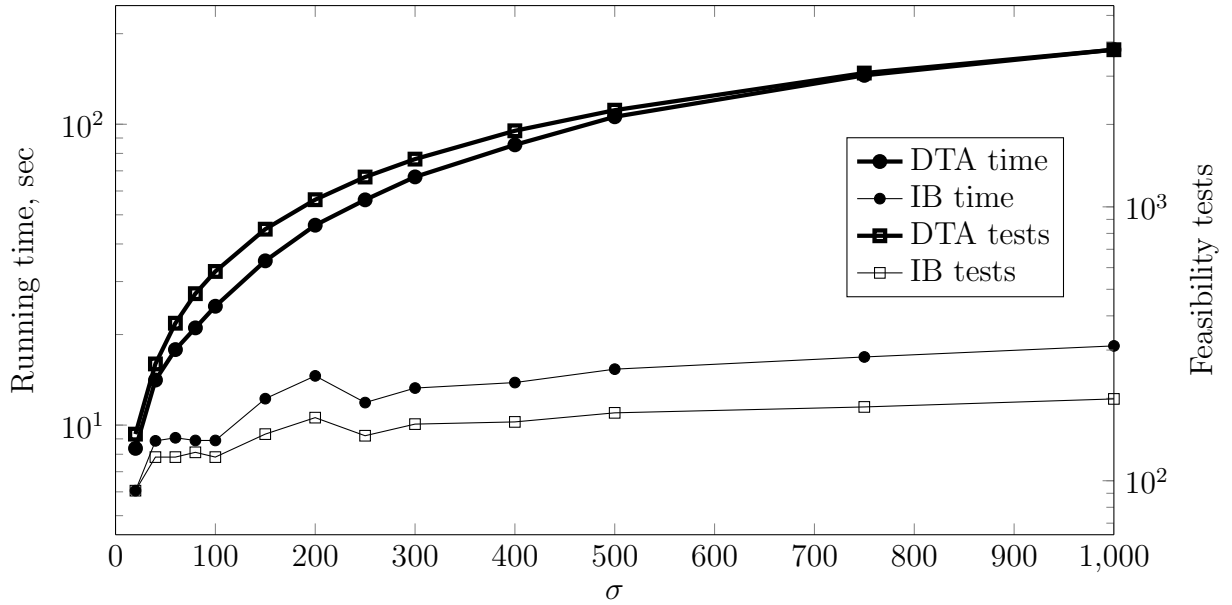


Figure 2: Analysis of the random instances with different values of parameter σ . The lines with circle marks indicate the running time of the algorithms and the lines with square marks indicate the number of feasibility tests applied within the algorithms.

proportional to p . In contrast, the IB algorithm efficiently handles instances with large p . Recall that it needs only $O(\log p)$ feasibility tests to solve each QBotKP subproblem [36]. Hence, for instances with large p , the IB algorithm appears preferable. Also it follows from our experiments that the time needed for the feasibility test is almost independent on the value of σ .

Based on these preliminary observations, we set $s = 0.1$ and $\sigma = 100$ for the rest of the experiments to assure that the test problems generated are reasonably hard.

6.2 Comparison of BDT, IB, DB and MIP

Let us first evaluate performance of the basic algorithms proposed in this paper. In Table 1, we report the results of our experiments with the BDT^B, BDT, IB^B, IB, DB^B, DB and MIP algorithms. The notations used in various columns of the table are explained below:

- m is the size of the test instance;
- ‘obj’ is the objective value of the optimal solution to the problem;
- p is the number of distinct c_{ij} values in matrix C ;
- $\Delta = \max_{i,j} c_{ij} - \min_{i,j} c_{ij}$;
- ‘Running time, sec’ columns report the running time of each of the algorithms;
- ‘Feasibility tests’ columns report the number of times feasibility tests are carried out within each of the algorithms. Note that in IB, IB^B, DB, and DB^B algorithms, we are not explicitly solving feasibility problems. However, feasibility problems of similar nature are solved within the QBotKP solver that is used within these algorithms. Thus, the number of feasibility tests include the feasibility tests carried out within the QBotKP solver used, which is a variation of the algorithm by Zhang and Punnen [36].

The last row of the table reports the average values for corresponding columns. However, it may be noted that the average running time can not be used to judge the algorithm’s performance in general because the running times vary significantly from instance to instance.

The best result (running time and number of tests) for each instance is underlined.

Feasibility test FT2 provides a better performance than feasibility test FT1 in each case. Indeed, according to Table 1, the optimality of a solution to the maximum weight independent set problem does not reduce the number of feasibility tests, while reaching the optimal solution clearly takes more time than finding a feasible solution.

The IB algorithm clearly outperforms all other algorithms for each test instance with regards to both the number of feasibility tests and the running time. The MIP algorithm turns out to be very slow for any practical instances.

m	obj	p	Δ	Running time, sec							Feasibility tests					
				BDT ^B	BDT	IB ^B	IB	DB ^B	DB	MIP	BDT ^B	BDT	IB ^B	IB	DB ^B	DB
50	64	453	689	2.7	2.2	0.6	<u>0.5</u>	0.6	0.5	1.7	463	462	94	<u>87</u>	97	95
100	239	568	728	131.5	22.0	56.2	<u>8.0</u>	99.3	12.2	7018.1	582	581	128	<u>125</u>	235	209
150	204	622	772	476.0	115.8	240.1	<u>37.5</u>	282.7	43.1	13187.2	642	643	191	<u>188</u>	227	228
200	181	672	778	938.1	217.7	475.3	<u>38.8</u>	756.0	75.1	—	684	684	117	<u>101</u>	207	174
250	206	701	807	3543.5	980.2	3806.5	<u>473.5</u>	8449.6	916.4	—	719	720	182	<u>178</u>	347	348
Avg.	179	603	755	1018.4	267.6	915.7	<u>111.7</u>	1917.6	209.5	—	618	618	142	<u>136</u>	223	211

Table 1: Comparison of basic algorithms.

m	obj	Ω	Running time, sec				Early		Iterations				Feasibility tests			
			BDT	BDT ^{Ω}	IB	IB ^{Ω}	BDT ^{Ω}	IB ^{Ω}	BDT	BDT ^{Ω}	IB	IB ^{Ω}	BDT	BDT ^{Ω}	IB	IB ^{Ω}
50	64	466	2.2	2.2	0.5	<u>0.4</u>	✓	✓	462	432	10	<u>9</u>	462	440	87	<u>86</u>
100	239	295	22.0	19.2	<u>8.0</u>	8.2	✓	✓	581	467	14	<u>13</u>	581	476	125	<u>124</u>
150	204	346	115.8	55.3	37.5	<u>34.5</u>	✓	✓	643	477	21	<u>20</u>	643	485	188	<u>186</u>
200	181	362	217.7	120.1	38.8	<u>37.1</u>	✓	✓	684	497	12	<u>11</u>	684	505	101	<u>99</u>
250	206	348	980.2	804.9	<u>473.5</u>	503.0	✓	✓	720	554	20	<u>19</u>	720	563	178	<u>177</u>
Avg.	179	363	267.6	200.3	<u>111.7</u>	116.6			618	485	15	<u>14</u>	618	494	136	<u>134</u>

Table 2: Analysis of efficiency of the early detection speed up.

6.3 Early Detection of Optimality

In this section we report the results of our experiments that assess the effectiveness of Theorem 1 for algorithms BDT and IB. In Table 2, we compare the results of the corresponding experiments. The ‘ Ω ’ column reports the objective value of the optimal solution to the QBP2 version of the bottleneck knapsack problem (QBotKP2):

$$\begin{aligned} & \text{Maximize } \min\{c_{ij} : (i, j) \in E \times E, x_i = x_j = 1\} - \min\{c_{ij} : (i, j) \in E \times E, x_i = x_j = 1\} \\ & \text{Subject to } \sum_{j=1}^m a_j x_j \geq b \end{aligned} \tag{9}$$

$$x_j \in \{0, 1\}, \text{ for } j = 1, 2, \dots, m. \tag{10}$$

This is used as the value of the parameter Ω in BDT $^\Omega$ and IB $^\Omega$ algorithms. The ‘Early’ columns report if early detection of optimality happened for the particular algorithm and test instance. ‘Iterations’ columns report the number of iterations of the algorithms.

In our experiments, early optimality detection happened for every test instance and every algorithm, where such an option was enabled. For the BDT algorithm, early detection significantly reduced the number of iterations and improved the overall performance. However, it reduces the number of iterations of the IB algorithm only by one in each run. Since calculating of Ω is approximately equivalent to one iteration of the IB algorithm, the overall running time of the procedure almost did not change after enabling early detection.

Another interesting observation that we can make from Table 2 is that the number of iterations of the IB algorithm almost does not depend on the size of the problem. Since each such iteration needs $O(\log p) = O(\log m)$ feasibility tests, the real running time of the IB algorithm per iteration is expected to be much higher than that of the BDT algorithm given large instances.

Recall that the value of Ω does not need to correspond to an optimal solution of the QBotKP2 but may be an upper bound to the optimal objective function value of QBotKP2. Observe that the algorithm used in our experiments to solve the QBotKP2 maintains a search interval which depends on the values of two indices ℓ, u where $\ell \leq u$. When $u - \ell < 1$ the algorithm terminates and guarantees an exact optimal solution [36]. This termination condition can be relaxed to $u - \ell < d$ to obtain a heuristic bound. In our implementation, we used the termination condition with $u - \ell < d$, where $1 \leq d \leq p$. Then the resulting upper bound $\Omega(d)$ can be calculated as $\Omega(d) = w_{u-1}$.

The results of our experiments with the approximate value of Ω are reported in Figure 3. When the value of d is relatively small, the upper bound $\Omega(d)$ is very close to the exact value of Ω . For $d > 40$, the upper bound becomes less accurate and the early detection happens in neither DTA $^{\Omega(d)}$ nor IB $^{\Omega(d)}$ algorithm. Thus, the effect of using an upper bound $\Omega(d)$ instead of the optimal objective value Ω is relatively small.

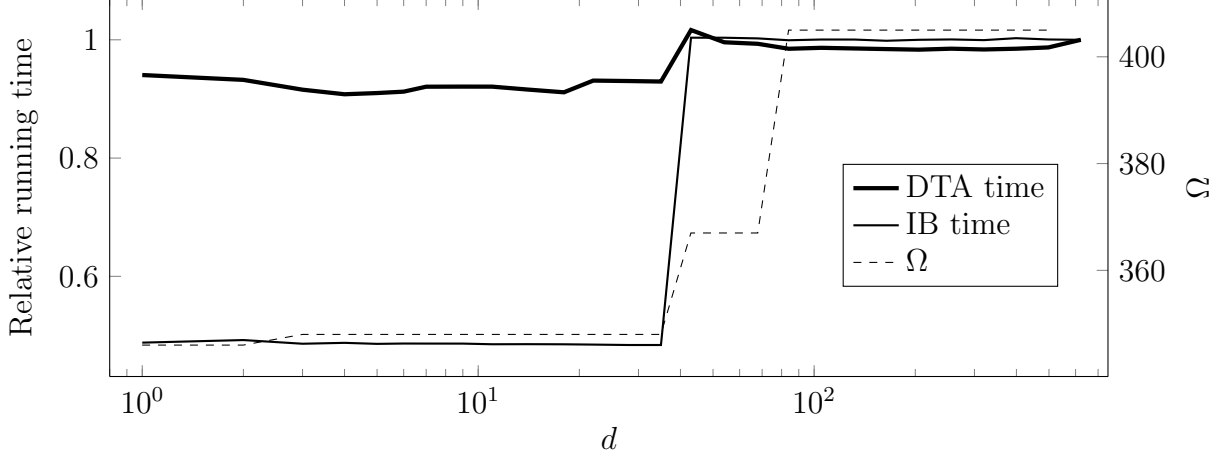


Figure 3: Analysis of the early optimality detection for different values of d . Observe that $\text{BDT}^{\Omega(1)} = \text{BDT}^{\Omega}$, and for $d = p$ we replace $\text{BDT}^{\Omega(p)}$ and $\text{IB}^{\Omega(p)}$ with the BDT and IB algorithms, respectively. The relative running time is calculated as t_d/t_1 , where t_d is the running time of the $\text{BDT}^{\Omega(d)}$ or $\text{IB}^{\Omega(d)}$ algorithm and t_1 is the running time of the BDT or IB algorithm, respectively. The experiments were conducted for a random instance of size $m = 150$.

6.4 Comparison of Heuristics

Recall that, by setting a time limitation to CPLEX when running a feasibility test, one can speed up the BDT, IB and DB algorithms at the cost of losing optimality guarantee. In Figure 4, we show how the solution quality depends on the running time for each of the BDT^t , IB^t and DB^t algorithms.

In our experiments, the running time and the solution quality of each of the heuristics monotonically depend on the parameter t . Thus, the balance between the solution quality and the running time in the proposed heuristics can be efficiently controlled by t . Note that the IB^t algorithm, likewise its exact version, shows the best performance among the proposed heuristics.

We also compared the BDT^t , IB^t and DB^t heuristics with the MIP algorithm given a time limitation (in this case, we set the ‘MIP Emphasis’ parameter of CPLEX to ‘Emphasize feasibility over optimality’). However, in our experiments, such a MIP heuristic showed very poor performance.

7 Conclusion

We introduced the combinatorial optimization model QBOP which can be used to model equitable distribution problems with pairwise interactions. The problem is strongly NP-hard even if the family of feasible solutions have a simple structure such as the collection of all subsets of a finite set with an upper bound on the cardinality of these subsets. Several exact

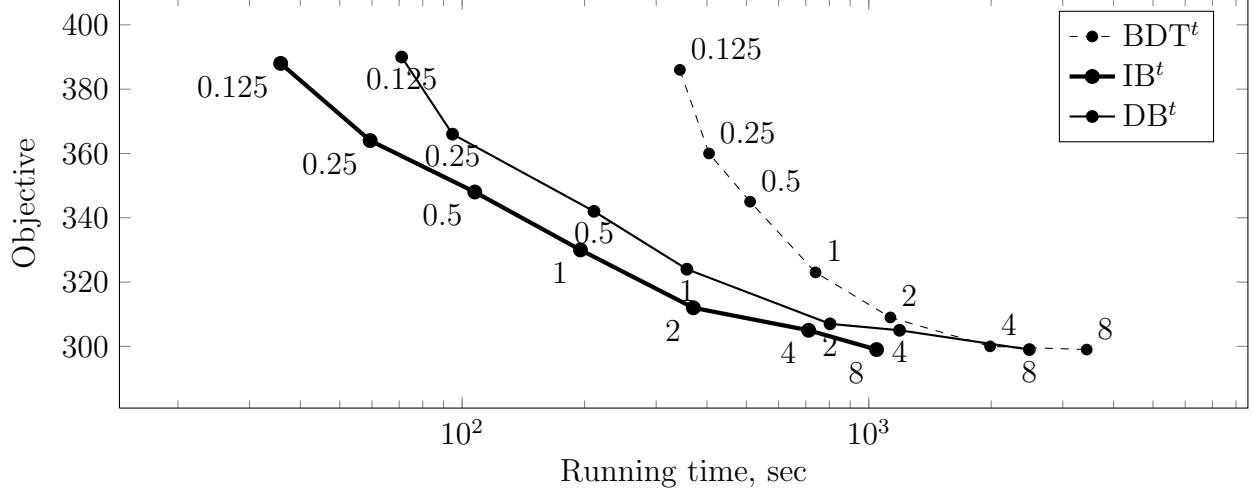


Figure 4: Performance of the BDT^t , IB^t and DB^t heuristics. The experiment is conducted for a random instance of size $m = 500$. The values of t , in seconds, are reported near each node.

and heuristic algorithms are provided along with detailed experimental analysis in the case of quadratic knapsack problems. Special cases of the problem with decomposable type cost matrices are discussed. It is shown that the complexity of the resulting QBOP depends on that of the corresponding LBOP.

By exploiting special structure of the family of feasible solutions \mathcal{F} and the structure of C , one may be able to obtain improved algorithms. These are interesting topics to investigate further, especially when real life situations warrant the study of such problems.

References

- [1] R.K Ahuja, Minimum cost to reliability ratio problem, *Computers and Operations Research* 15 (1988) 83–89.
- [2] R.K. Ahuja, The balanced linear programming problem, *European Journal of Operational Research* 101 (1997) 29–38.
- [3] Š. Berežný and V. Lacko, Balanced problems on graphs with categorization of edges, *Discussiones Mathematicae Graph Theory* 23 (2003) 5–21.
- [4] Š. Berežný and V. Lacko, Color-balanced spanning tree problems, *Kybernetika* 41 (2005) 539–546.
- [5] A. Brandstädt and V. Giakoumakis, Maximum weight independent sets in hole- and co-chair-free graphs, *Information Processing Letters* 112 (2012) 67–71.

- [6] A. Brandstädt, V.V. Lozin, and R. Mosca, Independent sets of maximum weight in apple-free graphs, *SIAM Journal on Discrete Mathematics* 24 (2010) 239–254.
- [7] R. E. Burkard, Quadratische bottleneckprobleme, *Operations Research Verfahren* 18 (1974) 26–41.
- [8] P. M. Camerini, F. Maffioli, S. Martello and P. Toth, Most and least uniform spanning trees, *Discrete Applied Mathematics* 15 (1986) 181–197.
- [9] P. Cappanera and M. G. Scutellà, Balanced paths in acyclic networks: Tractable cases and related approaches, *Networks* 45 (2005) 104–111.
- [10] Y. Dai, H. Imai, K. Iwano, N. Katoh, K. Ohtsuka, and N. Toshimura, A new unifying heuristic algorithm for the undirected minimum cut problem using minimum range cut algorithms, *Discrete Applied Mathematics* 65 (1996) 167–190.
- [11] A. Darmann, U. Pferschy, J. Schauer, G. J. Woeginger, Path, trees and matchings under disjunctive constraints, *Discrete Applied Mathematics* 159 (2011) 1726–1735.
- [12] C. W. Duin and A. Volgenant, Minimum deviation and balanced optimization: a unified approach, *Operations Research Letters* 10 (1991) 43–48.
- [13] D. Eppstein, Minimum Range Balanced Cuts via Dynamic Subset Sums, *Journal of Algorithms* 23 (1997) 375–385.
- [14] Z. Galil and B. Schieber, On finding most uniform spanning trees, *Discrete Applied Mathematics* 20 (1988) 173–175.
- [15] A. Grinèová, D. Kravecová, and M. Kulàè, Alternative approach to data network optimization, *Acta Electrotechnica et Informatica* 6 (2006) 1–5.
- [16] S. Gupta and T. Sen, Minimizing the range of lateness on a single machine, *Journal of the Operations Research society* 35 (1984) 853–857.
- [17] P. Hansen, G. Storchi and T. Vovor, Paths with minimum range and ratio of arc lengths, *Discrete Applied Mathematics* 78 (1997) 89–102.
- [18] N. Katoh and K. Iwano, Efficient algorithms for minimum range cut problems, *Networks* 24 (1994) 395–407.
- [19] J. LaRusic and A. P. Punnen, The balanced travelling salesman problem, *Computers & Operations Research* 38 (2011) 868–875.
- [20] C.-J. Liao and R.-H. Huang, An algorithm for minimizing the range of lateness on a single machine, *Journal of the Operational Research Society* 42 (1991) 183–186.
- [21] S. Martello, W. Pulleyblank, P. Toth and D. de Werra, Balanced optimization problems, *Operations Research Letters*, 3 (1984) 275–278.

- [22] E.Q.V. Martins, An algorithm to determine a path with minimal cost/capacity ratio, *Discrete Applied Mathematics* 8 (1984) 189–194.
- [23] T. Nemoto, An efficient algorithm for the minimum range ideal problem, *Journal of the Operations Research Society of Japan* 42 (1999) 88–97.
- [24] A.P. Punnen, On combined minmax-minsum optimization, *Computers and Operations Research* 21 (1994) 707–716.
- [25] A. P. Punnen and Y. P. Aneja, Lexicographic balanced optimization problems, *Operations Research Letters* 32 (2004) 27–30.
- [26] A. P. Punnen and K. P. K. Nair, Constrained balanced optimization problems, *Computers & Mathematics with Applications* 37 (1999) 157–163.
- [27] A.P. Punnen and R. Zhang, Quadratic bottleneck Problems, *Naval Research Logistics*, 58 (2011) 153–164.
- [28] M.G. Scutellà, A strongly polynomial algorithm for uniform balanced network flow problem, *Discrete Applied Mathematics* 81 (1998) 123–131.
- [29] M. Tegze and M. Vlach, Improved bounds for the range of lateness on a single machine, *Journal of the Operations Research society* 39 (1988) 675–680.
- [30] M. Tegze and M. Vlach, Minimizing maximum absolute lateness and range of lateness under generalized due dates, *Annals of Operations Research* 86 (1999) 507–526.
- [31] M. Tegze and M. Vlach, Minimizing the range of lateness on a single machine under generalized due dates, *INFOR* 35 (1997) 286–296.
- [32] L. Turner, A. P. Punnen, Y. P. Aneja, and H. W. Hamacher, On generalized balanced optimization problems, *Mathematical Methods of Operations Research* 73 (2011) 19–27.
- [33] L. Wu, An Efficient Algorithm for the Most Balanced Spanning Tree Problems, *Advanced science letters* 11 (2012) 776–778.
- [34] Z. Zeitlin, Minimization of maximum absolute deviation in integers, *Discrete Applied Mathematics* 3 (1981) 203–220.
- [35] R. Zhang, S. N. Kabadi and A. P. Punnen, The minimum spanning tree problem with conflict constraints and its variations, *Discrete Optimization* 8 (2011) 191–205.
- [36] R. Zhang and A.P. Punnen, Quadratic bottleneck knapsack problems, *Journal of Heuristics*, To appear.